

12-22-99

A

In the United States Patent and Trademark Office

Mailed 1999 December 20

Box Patent Application

Assistant Commissioner for Patents

Washington, District of Columbia 20231

Sir:

Please file the following enclosed patent application papers:

Applicant Name: Kendyl A. Román

Title: "VARIABLE GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (ZLN)"

☒ Specification, Claims, and Abstract: Nr. of Sheets 30

☒ Declaration: Date Signed: 1999 December 20

☒ Drawing(s): Nr. of Sheets Enc.: Formal: 13 Informal: _____

☐ Small Entity Declaration of Inventor(s) ☐ SED of Non-Inventor/Assignee/Licensee

☐ Assignment enclosed with cover sheet and recordal fee; please record and return.

☒ Check for \$ 467.00 for:

☒ \$ 467.00 for filing fee and additional claim fees (see attached Fee Transmittal)

☐ \$ _____ additional if Assignment is enclosed for recordal.

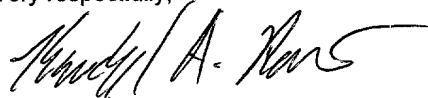
☐ Disclosure Document Program reference letter.

☒ Pursuant to 35 U.S.C. §119(e)(i), applicant(s) claim priority of Provisional Patent Application Ser. Nr. 60/113,051
filed 1998 December 21.

☒ Return Receipt Postcard Addressed to Applicant #1.

☒ Request Under MPEP § 707.07(j): The undersigned, a pro se applicant, respectfully requests that if the Examiner finds patentable subject matter disclosed in this application, but feels that Applicant's present claims are not entirely suitable, the Examiner draft one or more allowable claims for applicant.

Very respectfully,



Applicant Signature

730 Bantry Court, Sunnyvale, CA 94087-3402
Address (Send Correspondence Here)

Express Mail Label*

EK241178065US Date of Deposit 1999 December 20



09467721 12/20/99

Patent Application of

Kendyl A. Román

for

**TITLE: VARIABLE GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES
(ZLN)**

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority under 35 U.S.C. § 119(e) of the co-pending U.S. provisional application Serial Number 60/113,051 filed on 1998 December 21, and entitled "METHODS OF ZERO LOSS (ZL) COMPRESSION AND ENCODING OF GRAYSCALE IMAGES." The provisional application Serial Number 60/113,051 filed on 1998 December 21 and entitled "METHODS OF ZERO LOSS (ZL) COMPRESSION AND ENCODING OF GRAYSCALE IMAGES" is also hereby incorporated by reference.

A U.S. patent application to be filed on 1999 December 22, and entitled GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (RHN)", to be known as the "RHN" algorithm, has claims that combine some of the elements of the present invention in a different combination. The RHN application claims a priority date based on a co-pending U.S. provisional application Serial Number 60/113,276 filed on 1998 December 23.

BACKGROUND – FIELD OF THE INVENTION

This invention relates to data compression, specifically to the compression and decompression of video images.

BACKGROUND-DESCRIPTION OF PRIOR ART

In the last few years there have been tremendous advances in the speed of computer processors and in the availability of bandwidth of worldwide computer networks such as the Internet. These advances have led to a point where businesses and households now commonly have both the computing power and network connectivity necessary to have point-to-point digital communications of audio, rich graphical images, and video. However the transmission of video signals with the full resolution and quality of television is still out of reach. In order to achieve an acceptable level of video quality, the video signal must be compressed significantly without losing either spatial or temporal quality.

A number of different approaches have been taken but each has resulted in less than acceptable results. These approaches and their disadvantages are disclosed by Mark Nelson in a book entitled The Data Compression Book, Second Edition, published by M&T Book in 1996. Mark Morrisson also discusses the state of the art in a book entitled The Magic of Image Processing, published by Sams Publishing in 1993.

Video Signals

Standard video signals are analog in nature. In the United States, television signals contain 525 scan lines of which 480 lines are visible on most televisions. The video signal represents a continuous stream of still images, also known as frames, that are fully scanned, transmitted and displayed at a rate of 30 frames per second. This frame rate is considered full motion. A television screen has a 4:3 aspect ratio.

When an analog video signal is digitized each of the 480 lines are sampled 640 times, and each sample is represented by a number. Each sample point is called a picture element, or pixel. A two dimensional array is created that is 640 pixels wide and 480 pixels

high. This 640 x 480 pixel array is a still graphical image that is considered to be full frame. The human eye can perceive 16.7 thousand colors. A pixel value comprised of 24 bits can represent each perceivable color. A graphical image made up of 24-bit pixels is considered to be full color. A single second-long full frame, full color video requires over 220 millions bits of data.

The transmission of 640 x 480 pixels x 24 bits per pixel times 30 frames requires the transmission of 221,184,000 millions bits per second. A T1 Internet connection can transfer up to 1.54 millions bits per second. A high speed (56Kb) modem can transfer data at a maximum rate of 56 thousand bits per second. The transfer of full motion, full frame, full color digital video over a T1 Internet connection, or 56Kb modem, will require an effective data compression of over 144:1, or 3949:1, respectively.

A video signal typically will contain some signal noise. In the case where the image is generated based on sampled data, such as an ultrasound machine, there is often noise and artificial spikes in the signal. A video signal recorded on magnetic tape may have fluctuations due the irregularities in the recording media. Florescent or improper lighting may cause a solid background to flicker or appear grainy. Such noise exists in the real world but may reduce the quality of the perceived image and lower the compression ratio that could be achieved by conventional methods.

Basic Run-length Encoding

An early technique for data compression is run-length encoding where a repeated series of items are replaced with one sample item and a count for the number of times the sample repeats. Prior art shows run-length encoding of both individual bits and bytes. These simple approaches by themselves have failed to achieve the necessary compression ratios.

Variable Length Encoding

In the late 1940s, Claude Shannon at Bell Labs and R.M. Fano at MIT pioneered the field of data compression. Their work resulted in a technique of using variable length codes where codes with low probabilities have more bits, and codes with higher probabilities have fewer bits. This approach requires multiple passes through the data to determine code probability and then to encode the data. This approach also has failed to achieve the necessary compression ratios.

D. A. Huffman disclosed a more efficient approach of variable length encoding known as Huffman coding in a paper entitled “A Method for Construction of Minimum Redundancy Codes,” published in 1952. This approach also has failed to achieve the necessary compression ratios.

Arithmetic, Finite Context, and Adaptive Coding

In the 1980s, arithmetic, finite coding, and adaptive coding have provided a slight improvement over the earlier methods. These approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

Dictionary-Based Compression

Dictionary-based compression uses a completely different method to compress data.

Variable length strings of symbols are encoded as single tokens. The tokens form an index to a dictionary. In 1977, Abraham Lempel and Jacob Ziv published a paper entitled, “A Universal Algorithm for Sequential Data Compression” in IEEE Transactions on Information Theory, which disclosed a compression technique commonly known as LZ77. The same authors published a 1978 sequel entitled, “Compression of Individual Sequences via Variable-Rate Coding,” which disclosed a compression technique commonly known as LZ78 (see U.S. Patent 4,464,650). Terry Welch published an article entitled, “A Technique for High-Performance Data Compression,” in the June 1984 issue of IEEE Computer, which disclosed an algorithm commonly known as LZW, which is the basis for the GIF algorithm

(see US Patents 4,558,302, 4,814,746, and 4,876,541). In 1989, Stack Electronics implemented a LZ77 based method called QIC-122 (see U.S. Patent 5,532,694, U.S. Patent 5,506,580, and U.S. Patent 5,463,390).

These lossless (method where no data is lost) compression methods can achieve up to 10:1 compression ratios on graphic images typical of a video image. While these dictionary-based algorithms are popular, these approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

JPEG and MPEG

Graphical images have an advantage over conventional computer data files: they can be slightly modified during the compression/decompression cycle without affecting the perceived quality on the part of the viewer. By allowing some loss of data, compression ratios of 25:1 have been achieved without major degradation of the perceived image. The Joint Photographic Experts Group (JPEG) has developed a standard for graphical image compression. The JPEG lossy (method where some data is lost) compression algorithm first divides the color image into three color planes and divides each plane into 8 by 8 blocks, and then the algorithm operates in three successive stages:

- (a) A mathematical transformation known as Discrete Cosine Transform (DCT) takes a set of points from the spatial domain and transforms them into an identical representation in the frequency domain.
- (b) A lossy quantization is performed using a quantization matrix to reduce the precision of the coefficients.
- (c) The zero values are encoded in a zig-zag sequence (see Nelson, pp. 341-342).

JPEG can be scaled to perform higher compression ratio by allowing more loss in the quantization stage of the compression. However this loss results in certain blocks of the image being compressed such that areas of the image have a blocky appearance and the edges of the 8 by 8 blocks become apparent because they no longer match the colors of their

adjacent blocks. Another disadvantage of JPEG is smearing. The true edges in an image get blurred due to the lossy compression method.

The Moving Pictures Expert Group (MPEG) uses a combination of JPEG based techniques combined with forward and reverse temporal differencing. MPEG compares adjacent frames and for those blocks that are identical to those in a previous or subsequent frame and only a description of the previous or subsequent identical block is encoded. MPEG suffers from the same blocking and smearing problems as JPEG.

These approaches require extensive computer processing and have failed to achieve the necessary compression ratios without unacceptable loss of image quality and artificially induced distortion.

QuickTime: CinePak, Sorensen, H.263

Apple Computer, Inc. released a component architecture for digital video compression and decompression, named QuickTime. Any number of methods can be encoded into a QuickTime compressor/decompressor (codec). Some popular codec are CinePak, Sorensen, and H.263. CinePak and Sorensen both require extensive computer processing to prepare a digital video sequence for playback in real time; neither can be used for live compression. H.263 compresses in real time but does so by sacrificing image quality resulting in severe blocking and smearing.

Fractal and Wavelet Compression

Extremely high compression ratios are achievable with fractal and wavelet compression algorithms. These approaches require extensive computer processing and generally cannot be completed in real time.

Sub-sampling

Sub-sampling is the selection of a subset of data from a larger set of data. For example, when every other pixel of every other row of a video image is selected, the resulting image has half the width and half the height.

Image Stretching

If an image is to be enlarged but maintain the same number of pixels per inch, data must be filled in the new pixels that are added. Various methods of stretching an imaging and filling in the new pixels to maintain image consistency. Some methods known in the art are dithering (using adjacent colors that appear to be blended color), and error diffusion, “nearest neighbor”, bilinear and bicubic.

SUMMARY OF THE INVENTION

In accordance with the present invention a method of compression of a video stream comprises steps of sub-sampling a video frame, and run-length encoding the sub-sampled pixel values, whereby the method can be executed in real time and the compressed representation of pixels saves substantial space on a storage medium and require substantially less time and bandwidth to be transported over a communications link. The present invention includes a corresponding method for decompressing the encoded data.

Objects and Advantages

Accordingly, beside the objects and advantages of the method described in our patent above, some additional objects and advantages of the present invention are:

- (a) to provide a method of compressing and decompressing video signals so that the video information can be transported across a digital communications channel in real time.
- (b) to provide a method of compressing and decompressing video signals such that compression can be accomplished with software on commercially available computers without the need for additional hardware for either compression or decompression.
- (c) to provide a high quality video image without the blocking and smearing defects associated with prior art lossy methods.

- (d) to provide a high quality video image that suitable for use in medical applications.
- (e) to enhance images by filtering noise or recording artifacts.
- (f) to provide a method of compression of video signals such that the compressed representation of the video signals is substantially reduced in size for storage on a storage medium.
- (g) to provide a level of encryption so that images are not directly viewable from the data as contained in the transmission.

DRAWING FIGURES

In the drawings, closely related figures have the same number but different alphabetic suffixes.

Fig 1 shows the high level steps of compression and decompression of an image.

Fig 2A to 2H show alternatives for selecting a pixel value for encoding.

Fig 3A shows the variable encoding format.

Fig 3B shows an example of a code where N is 5 bits wide and U is 3 bits wide.

Fig 4A shows the flowchart for the compression method.

Fig 4B shows an image and a corresponding stream of pixels.

Fig 5A to 5C shows the formats for the run-length encoding of the RHN algorithm.

Fig 6 shows a series of codes and the resulting encoded stream.

Fig 7 shows a series of codes and the resulting encoded stream of the RHN algorithm.

Fig 8A shows examples of variable formats.

Fig 8B shows a format that preserves 9 bits of color.

Fig 9 shows the flow chart for the decompression method.

Figs 10 shows image stretching by interpolation.

Figs 11A and 11B show an encryption table and a decryption table.

Reference Numerals in Drawings

100	compression steps	110	sub-sampling step
130	encoding step		
140	encoded data	150	decompression steps
160	decoding step	180	image reconstitution step
200	32 bit pixel value		
202	blue channel	204	green channel
206	red channel	208	alpha channel
210	24 bit pixel value	212	blue component
214	green component	216	red component
220	RGB averaging diagram	222	blue value
224	green value	226	red value
228	averaged value	230	blue selection diagram
232	blue instance	234	green instance
236	red instance	240	selected blue value
250	green selection diagram	260	selected green value
270	red selection diagram	280	selected red value
290	grayscale pixel	292	grayscale blue
294	grayscale green	296	grayscale red
298	selected grayscale value	299	filtered pixel value
300	N	301	U
302	W	310	pixel bit 7
312	pixel bit 6	314	pixel bit 5
316	pixel bit 4	318	pixel bit 3
320	pixel bit 2	322	pixel bit 1
324	pixel bit 0	325	8 bit pixel
330	5 bit sample		

10

332	sample bit 4	334	sample bit 3
336	sample bit 2	338	sample bit 1
340	sample bit 0	350	3 low order bits
360	formatted code	380	3 bit count value
400	encode flowchart		
402	encode entry	403	encode initialization step
404	get pixel step	405	get value step
406	lookup encoded value step	408	compare previous
410	increment counter step	412	check count overflow
414	new code step	416	check end of data
418	set done	420	counter overflow step
422	check done	428	encode exit
430	image	440	image width
450	image height	460	pixel stream
500	code byte	510	flag bit
520	repeat code	530	count
550	data code	560	wasted bits
565	data bit 6		
570	data bit 5	575	data bit 4
580	data bit 3	585	data bit 2
590	data bit 1	595	data bit 0
610	decimal values	620	first value
622	second value	624	third value
626	fourth value	628	fifth value
630	sixth value	632	seventh value
640	binary code	650	first byte

651	first data	652	first count
653	second byte	654	second data
655	second count	656	third byte
657	third data	658	third count
740	RHN binary code	803	ZL3 format
804	ZL4 format	805	ZL5 format
808	ZL8 format	809	ZL9 format
812	ZL12 format	820	ZL9C format
900	decode entry	901	decode initialize step
902	get code step	908	decode lookup step
909	check zero count	910	place pixel step
914	reset counter step	916	check length
918	decode exit	1000	encryption key
1010	first adjacent pixel	1012	second adjacent pixel
1010	first subsequent adjacent pixel	1012	second subsequent adjacent pixel
1052, 1054, 1056, 1058, 1060			interpolated pixels
1100	encryption table	1110	decryption table

DESCRIPTION OF THE INVENTION

Fig 1–Compression and Decompression Steps

Fig 1 illustrates a sequence of compression steps 100 and a sequence of decompression steps 150 of the present invention. The compression steps 100 comprise a sub-sampling step 110 and an encoding step 130. After completion of the compression steps 100, a stream of encoded data 140 is output to either a storage medium or a transmission channel. The decompression steps 150 comprise a decoding step 160 wherein the stream of encoded data 140 is processed and an image reconstitution step 180.

Figs 2A to 2H Selecting Pixel Values for Encoding

Figs 2A to 2G illustrate alternatives for selecting a pixel value for encoding. The sub-sampling step 110 (Fig 1) includes sub-sampling of a pixel value to obtain a variable selected number of bits.

Video digitizing hardware typical has the options of storing the pixel values as a 32 bit pixel value 200 or a 24 bit pixel value 210, shown in Fig 2A and Fig 2B, respectively. The 32 bit pixel value 200 is composed of a blue channel 202, a green channel 204, a red channel 206, and an alpha channel 208. Each channel contains of 8 bits and can represent 256 saturation levels for the particular color channel. For each channel the saturation intensity value of zero represents the fully off state, and the saturation intensity value of “255” represents the fully on state. A common alternative not shown is a sixteen bit format where the three color channels contain 5 bits each and the alpha channel is a single bit. The present invention anticipates the use of the color channels of 16 bit pixel value in a manner substantially the same as the 32-bit pixel value 200 except the number of bits per channel is 5 instead of 8.

The 24 bit pixel value 210 is composed of a blue component 212, a green component 214, and a red component 216. There is no component for the alpha channel in the 24 bit pixel value 210. Regardless of the structure, the blue channel 202 is equivalent to the blue component 212, the green channel 204 is equivalent to the green component 214, and the red channel 206 is equivalent to the red component 216.

In the present invention, the 32 bit pixel value 200 alternative is preferred due to the consistent alignment of 32 bit values in most computer memories; however for simplicity of illustration the alpha channel 208 will be omitted in Fig 2C to 2G.

If the video signal is digitized in color, the three color components may have different values. For example in Fig 2C, a RGB averaging diagram 220 illustrates a blue value 222 of 35 decimal, a green value 224 of 15, and a red value 226 of 10. One alternative is to sub sample from 24 bits to 8 bits by averaging the three color values to obtain an

averaged value 228 that, in this example, has the value of 20. $(10+15+35)/3 = 20$. This will produce a grayscale image. Alternatively, a color image can be preserved by sampling bits from each color component (see Fig 8B).

Fig 2D illustrates another alternative for selecting an 8 bit value in a blue selection diagram 230. In this example, a blue instance 232 has the value of 35, a green instance 234 has the value of 15, and a red instance 236 has the value of 10. In this alternative the blue instance 232 is always selected as a selected blue value 240.

Fig 2E illustrates another alternative for selecting an 8 bit value in a green selection diagram 250. In this alternative the green instance 234 is always selected as a selected green value 260.

Fig 2F illustrates another alternative for selecting an 8 bit value in a red selection diagram 270. In this alternative the red instance 236 is always selected as a selected red value 280.

If the video signal being digitized is grayscale, the three color components will have the same values. For example in Fig 2G, a grayscale pixel 290 comprises a grayscale blue 292 with a value of decimal 40, a grayscale green 294 with a value of 40, and a grayscale red with a value of 40. Because the values are all the same, it makes no difference which grayscale color component is selected, a selected grayscale value 298 will have the value of 40 in this example.

The preferred embodiment of this invention uses the low order byte of the pixel value, which is typically the blue component as shown in Fig 2D.

Fig 2H illustrates a filtered pixel value 299 of 8 bits that may be selected by one of the alternatives described above. In these examples, the filtered pixel value 299 is equivalent to items referenced by numerals 228, 240, 260, 280, or 298. This reduction of the 32 bit pixel value 200 or the 24 bit pixel value 210 contributes a reduction in data size of 4:1

or 3:1, respectively. This reduction recognizes that for some images, such as medical images or grayscale images, no relevant information is lost.

For additional compression, the filtered pixel value 299 can variably select any number of bits. For example, selection of the most significant four bits instead of all eight bits filters noise that may show up in the low order bits may be very suitable for an image such as one produced by an ultrasound medical device. An example of this is shown by ZL4 804 in Fig 8A.

Figs 3A and 3B—Encoding Formats

Speed of compression and decompression may be enhanced if the algorithms fit into computer memory native storage elements such as 8 bit bytes, 16 bit words, or 32 bit double words, or some other size for which the computer architecture is optimized.

A grayscale image may be stored at a higher bit level than the actual values require. This may occur when an image is generated by an imaging technology such as radar, ultrasound, x-ray, magnetic resonance, or similar electronic technology. For example an ultrasound machine may only produce 16 levels of grayscale, requiring 4 bits of data per pixel, but the image digitizing may be performed at 8 to 12 bits per pixel. In this example, the low order bits (4 to 8) respectively provide no significant image data.

In the present invention, a fast and efficient compression and encoding algorithm is implemented by using unused bits to store a repeat count for repeated values.

The most significant N bits of the pixel value are selected where N is the number of significant bits (determined by data analysis or by user selection). If N is less than W, where W is a native machine data type such as 8 bit byte, 16 bit word, or 32 bit double word or some other size for which the computer architecture is optimized, then W-N equals the number of unneeded bits, U. A repeat count, C, can contain a value from 1 to CMAX where CMA is 2 to the power of U. For example, if U equals 4, C can be a number from 1 to 16.

In practice the maximum value will be encoded as a zero because the high order bit is truncated. In the example, decimal 16 has a binary value "10000" will be stored as "0000".

For example, when W is 8, value pairs for N and U could include without limitation (2,6), (3,5), (4,4), (5,3), and (6,2). When W is 16, value pairs for N and U could include without limitation (2,14), (3,13), (4,12), (5,11), (6,10), (7, 9), (8, 8), (9, 7), (10, 6), (11, 5), (12, 4), (13, 3), and (14, 2). When W is 32, value pairs for N and U could include without limitation all combinations of values pairs for N and U where $N+U$ equals 32 and $N > 1$ and $U > 1$. When W is not a multiple of 8, value pairs for N and U could include without limitation all combinations of values pairs for N and U where $N+U$ equals W and $N > 1$ and $U > 1$.

Figs 3A shows the encoded format where N 300 represent the N most significant bits of the pixel value 299, U 301 represents the bits that are not used for the data and are therefor used for the repeat count, and W 302 where W is the width of the encoded data and equal to sum of N and U. As stated above W is preferably a native machine element.

Fig 3B illustrates bit sub-sampling where N's 300 bit width is 5, U's 301 bit width is 3, and W 302 is 8. The high order 5 bits 310-318 of an 8 bit pixel 325 are extracted to form a five bit sample 330. The lower 3 bits of 330 are ignored bits 350. In the formatted code 360, the ignored bits 350 are replaced with the repeat count value 380.

Encoding

The most significant N bits of each pixel are selected from the image to obtain value V.

In the encryption embodiment of this invention V may be used to select an encoded value, E, from the encoding table. E is also a N-bit value. The number of elements in the encode table 1100 (Fig 11) is 2 to the Nth power.

In the other embodiments of this invention V is used as E.

E is saved as the prior value, P. For each subsequent pixel, the encoded value, E, is obtained and compared to the prior value, P. If the prior value, P, is the same as E, then a repeat counter, C, is incremented; otherwise the accumulated repeat count, C, for the prior value, P, is merged with P and placed in an array A that implements the encoded data 140 (Fig 1) buffer. For example, if W is 8 and N is 4 and C is 10, U is 4, CMAX is 16, and $((P \ll U) \mid C)$ is the merged value. If the repeat count, C, is greater CMAX, then CMAX is merged with P $((P \ll U) \mid CMAX)$ and placed in the encoded data 140 (Fig 1) buffer, A. CMAX is subtracted from C and merged values are placed in A until C is less than CMAX. All pixels are processed in this manner until the final value is compressed and encoded. The length, L, of the encoded data 140 (Fig 1) is recorded with the encoded data 140.

Fig 4A—Encode Flowchart

Fig 4A illustrates the encode flowchart 400 which represents the details of the encryption embodiment of the encoding step 130 (Fig 1) for the present invention.

The encoding begins at an encode entry 402. In an encode initialization step 403, a prior value P is set to a known value, preferably decimal “255” or hexadecimal 0xFF, a repeat counter C is set to zero, an encoded length L is set to 0, and a completion flag “Done” is set to a logical value of false. Next, a get pixel step 404 obtains a pixel from the image being encoded. At a get value step 405, a value V is set to the N bit filter pixel value 299 as derived from the pixel using one of the methods shown in Fig 2C to 2G, preferably the fastest as explained above, and extracting the N most significant bits. At a lookup encoded value step 406, an encoded value E is set to the value of one of the codes 1105 (Fig 11A) of the encode table 1100 as indexed by V. (In the non encrypted embodiment of this invention, step 406 is bypassed because V is used as E) Next, a compare previous 408 decision is made by comparing the values of E and P. If the values are the same, an increment counter step 410 is executed and flow continues to the get pixel step 404 that obtains the next pixel from the image.

If the encode value E does not match the prior value P, then a check count overflow 412 decision is made. If the counter C is less than or equal to CMAX, then a new code step 414 is executed, otherwise a counter overflow step 420 is executed.

At step 414, the counter C is masked and bit-wise OR-ed with P shifted left by U bit positions and is placed in the A at the next available location as indexed by the encoded length L. Then, continuing inside flowchart step 414, L is incremented, the repeat count C is set to 1 and the prior value P is set to E. After step 414, a check end of data decision is made by checking to see if there are any more pixels in the image and if not if the last value is has been processed. Because this method utilizes a read ahead technique step 414 must be executed one more time after the end of data is reached to process the last run-length. If there is more data in the image, flow continues to a check of the completion flag "Done" at step 422. If the check indicates that the process is not completed, flow continues to step 404.

If the end of data is reached but the completion flag "Done" is still false, flow continues to a set done step 418. At step 418, the completion flag "Done" is set to logical true, and flow continues to decision 412 where the last run-length will be output and flow will eventually exit through step 414, decision 416, decision 422, and then terminate at encode exit 428.

It is possible for the repeat count C to become larger than CMAX requiring more bits than allocated by this method. This situation is handled by making the check count overflow 412 decision and executing the counter overflow step 420. At step 420, the counter C is masked and bit-wise OR-ed with P shifted left by U bit positions and is placed in the A at the next available location as indexed by the encoded length L. Then, continuing inside flowchart step 414, L is incremented, the repeat count C is decrement by CMAX. After step 420, flows continues to the check count overflow 412 decision. Thus when the

encode value E repeats more that CMAX times, multiple sets of repeat counts and encoded values are output to the encoded data 140 buffer.

This entire process is repeated for each image or video frame and the encoded length L is transmitted with the encoded data associated with each frame. The encoded length varies from frame to frame depending on the content of the image being encoded.

Fig 4B–Image and Pixel Stream

Fig 4B illustrates an image and its corresponding stream of pixels. A rectangular image 430 is composed of rows and columns of pixels. The image 430 has a width 440 and a height 450, both measured in pixels. Pixels in a row are accessed from left to right. Rows are accessed from top to bottom. Some pixels in the image are labeled from A to Z. Pixel A is the first pixel and pixel Z is the last pixel. Scanning left to right and top to bottom will produce a pixel stream 460. In the pixel stream 460, pixels A and B are adjacent. Also pixels N and O are adjacent even though they appear on different rows in the image. If adjacent pixels have the same code the process in Fig 4A will consider them in the same run.

Because the video signal being digitized is analog there will be some loss of information in the analog to digital conversion. The video digitizing hardware can be configured to sample the analog data into the image 430 with almost any width 440 and any height 450. The present invention achieves most of its effective compression by sub-sampling the data image with the width 440 value less than the conventional 640 and the height 450 value less than the convention 480. The preferred embodiment of the invention for use in a medical application with T1 Internet transmission bandwidth is to sample at 320 by 240. However a sampling resolution of 80 by 60 may be suitable for some video application.

Figs 5A to 5C—Run-length Encoding Formats of the RHN Algorithm

Figs 5A to 5C show use of a different structure than the present invention. Figs 5A to 5C show the formats for the run-length encoding of the RHN. In Fig 5A, a code byte 500, with its high order bit designated as a flag bit 510.

Fig 5B shows a repeat code 520 comprising a Boolean value one in its flag bit 510 and a 7 bit count 530 in the remaining 7 low order bits. The seven bit count 530 can represent 128 values with a zero representing “128” and 1 through 127 being their own value.

Fig 5C shows a data code 550 comprising:

1. a Boolean value zero in its flag bit 510
2. two unused data bits: data bit 6 reference by 565 and data bit 5 reference by 570, and
3. five bits, data bits 4 to 0, reference by 575, 580, 585, 590, and 595, respectively.

Fig 5C shows that in every byte of the RHN data code 550 two bits are unused and one bit is used for the flag bit, so that only five of the eight bits are used for data. The remaining three bits are wasted bits 560. The present invention uses a different structure by placing the repeat count in bits that the RHN format would not have used for data (U). The corresponding ZLN format, ZL5 (where N is 5, U is 3, and W is 8), always uses five bits for data and the remaining 3 bits for the repeat count. In practice, repeat counts are small and often can fit in 3 bits, so this embodiment of the present invention will result in superior compression performance over the RHN algorithm.

In addition, the present invention provides for a larger count when the bit filtering is larger. For example, the alternate ZLN format where each byte contains 4 data bits, ZL4 (where N is 4 and U is 4), allows for a four bits of repeat count. For example, in practice, ZL4 is superior to RHN on a typical ultrasound image containing 16 shades of gray.

Fig 6–Encoded Data Stream

Fig 6 shows a series of decimal values 610 comprising a first value 620 equal to decimal 0, a second value 622 equal to 0, a third value 624 equal to 0, a fourth value 626 equal to 0, a fifth value 628 equal to 0, a sixth value 630 equal to 2, and a seventh value 632 equal to 10. After the encoding step 130 (Fig 1), the corresponding encoded data 140 (Fig 1) would be compressed down to three bytes of binary code 640 comprising a first byte 650, a second byte 653, and a third byte 656 each containing a merged value and count, (651, 652), (654, 655), and (657, 658), respectively. The first data 651 has a binary value of “00000” which equals the repeated decimal value zero. The first count 652 has a binary value “101” which equals decimal five representing the run-length of the repeating value in the first five of the decimal values 610. The second data 654 has a binary value of “00010” which equals the non-repeated decimal value two. The second count 655 has a value of 1. The third data 657 has a binary value of “01010” which equals the non-repeated decimal value ten. The third count 658 has a value of 1.

Fig 7–RHN Codes and Encoded Stream

Fig 7 shows the same series of decimal values 610 (Fig 6) comprising the first value 620 equal to decimal 0, the second value 622 equal to 0, the third value 624 equal to 0, the fourth value 626 equal to 0, the fifth value 728 equal to 0, the sixth value 730 equal to 2, and the seventh value 732 equal to 10. After encoding by RHN, the corresponding encoded data 140 (Fig 1) would be compressed down to four bytes of RHN binary code 740.

The embodiment of the present invention shown in Fig 6 only requires three bytes to encode the same data. In this example, the present invention is 25% better than the ZHN format.

Figs 8A and 8B–ZLN Formats

The ZLN algorithm of the present invention provides for variable formats. The values of N 300, U 301, and W 302 can be dynamically changed between frames. For ease

of communication a format is named with the prefix “ZL” and a digit representing the value of N. For example, “ZL5” refers to a format where bit width of N is equal to 5. There are multiple values of U depending of the W. To also specify the bit width of U a hyphen and a number can be appended. For example, “ZL5-13” represents a format where $N = 5$ and $U = 13$. “ZL5-8” is a common format and may be imprecisely referred to as “ZL5.”

Figure 8A shows a number of formats with adjacent labels: ZL3 803, ZL4 804, ZL5 805, ZL8 808, ZL9 809, and ZL12 812. Data bits are represented by “D,” and count bits are represented by “C”.

Figure 8B shows how the most significant 3 bits of each color component (216, 214, and 212 of Fig 2B) are extracted and formatted in ZL9-7C format (the “C” append indicates that the color is preserved). With three red bits represented by “R”, three green bits represented “G” and three blue bits represented by “B”.

Decoding

To decode the compressed array, the decoder has a decode table that corresponds with the encode table. For $W*4$ bit color pixels, the decode table contains the appropriate alpha, red, green, and blue values. For $W*3$ bit color pixels, the alpha value is not used. The compressed array is processed W bits at a time as X. The repeat count, C, is extracted from X by masking off the data value ($C = X \& (((2**N)-1) \ll U)$). The encoded value, E, is extracted from X by masking off the count ($E = X \& ((2**U)-1)$). The encoded value, E maybe used to index into the decryption. The decoded pixels are placed in a reconstructed image and repeated C times. Each element of the compressed array, A, is processed until its entire length, L, has been processed.

Fig 9–Decode Flowchart

Fig 9 illustrates the decode flowchart which presents the details of the decryption embodiment of the decode step 160 (Fig 1) and the image reconstitution step 180 (Fig 1).

The decoding begins at a decode entry 900. In a decode initialization step 901, a repeat counter C is set to one, an encoded length L is set to the value obtained with the encoded data 140 (Fig 1), and an index I is set to 0. Next, a get code step 902 obtains a signed byte X from the encoded data 140 (Fig 1) array A. The index I is incremented. The count #XX (Fig 3X) is extracted from X by masking off the data bits and placed in the repeat counter C ($C = X \& ((2^{*N}) - 1 \ll U)$). The value of E is extracted from X by masking off the count bits ($E = X \& (2^{*U} - 1)$). In practice, the count mask and value mask can be pre-computed with the following two lines of code in the C programming language:

```
valueMask = -1 << U;
countMask = ~valueMask;
```

In the decryption embodiment of the present invention flow goes to a decode lookup step 908 where the value of E is used to index into the decode table 1110 (Fig 11) to obtain a pixel value V. In the other embodiments where E is not encrypted, E is used as V and step 908 is bypassed. Flow continues to a check zero count 909 decision.

The 909 decision always fails the first time ensuring that a place pixel step 910 is executed. The place pixel step 910 places the pixel value V in the next location of the decompressed image and decrements the repeat counter C and returns to the 909 decision. The pixel value V is placed repeatedly until C decrements to zero. Then the 909 decision branches flow to a reset counter step 914. At step 914 the repeat counter is reset to 1.

Flow continues to the check length 916 decision where the index I is compared to the encoded length L to determine if there are more codes to be processed. If I is less than L flow returns to step 902, otherwise the decode process terminates at a decode exit 918.

The entire decode process is repeated for each frame image.

Fig 10–Interpolation

Fig 10 interpolation when a two adjacent pixels 1010 and 1012 and two subsequent row adjacent pixels 1014 and 1016 are stretched to insert a new row and column of pixels.

Pixels 1052, 1054, 1056, 1058 and 1060 are insert due to the enlargement of the image. Their values are calculated by averaging the values of the two pixels above and below or to the left or the right of the new pixel. A preferred sequence is calculation of:

1. 1052 between 1010 and 1012
2. 1054 between 1010 and 1014
3. 1058 between 1012 and 1016
4. 1056 between 1054 and 1058

Pixel 160 can be calculated on the interpolation for the subsequent row.

Fig 11–Encryption

By using corresponding encoding and decoding tables the data can be encrypted and decrypted without using actual values. Encryption provides a level of security for the encoded data 140 while in storage or transit.

Fig 11 shows an example of an encryption table 1100, where N is 3 and W is 8, and a decryption table 1110, where N is 3 and U is 5.

The encode table 1100 is 2 the power of N in length. If the target color image format is W*4 bit color, then the decode table 1110 has W bits for alpha, red, green, and blue each, respectively. If the target color image format is W*3 bit color, then the alpha value is not used. If the image is W bit grayscale then only the grayscale value is used to create the decompressed and decoded image.

The corresponding table elements are mapped to each other. For example, 0 could encode to 22 as long as the 22nd element of the decode table returns $(0xff \ll 24 \mid 0 \ll 16 \mid 0 \ll 8 \mid 0)$.

When these versions of the tables are used, the encode and decode processes and their speed of execution are substantially the same but the encoded data 140 (Fig 1) becomes a cipher and has a higher level of security. It should be recognized by one with ordinarily

skill in the art that there are other embodiments of the present invention with different encryption/decryption table rearrangements.

Advantages

Noise Filtering and Image Enhancement

The removal of the least significant bits of pixel values results in high quality decompressed images when the original image is generated by an electronic sensing device such as an ultrasound machine which is generating only a certain number of bits of grayscale resolution. By variably altering the number of most significant bits various filters can be implemented to enhance the image quality. Such a noise filter can be beneficial when the image is generated by an imaging technology such as radar, ultrasound, x-ray, magnetic resonance, or similar technology. Variations can be made to enhance the perceived quality of the decompressed image. Therefore, altering the number of data bits selected and altering the width of the repeat count is anticipated by this invention and specific values in the examples should not be construed as limiting the scope of this invention.

Dynamic Variable Formats

While a video stream is being viewed a viewer on the decoding end of the transmission can vary the settings for the compressor. Different tradeoffs between image spatial and temporal quality can be made. As the contents of the video signal change an appropriate format can be selected. Control signals can be sent back to the compressor via a communications link.

Execution Speed

The preferred embodiment of this invention use a number of techniques to reduce the time required to compress and decompress the data.

The methods require only a single sequential pass through the data. Both the compression steps 100 and the decompression steps 150 access a pixel once and perform all calculations.

When selecting the filtered pixel value 299, the preferred embodiment selects the low order byte from the 32 bit pixel value 200 or the 24 bit pixel value 210 so that an additional shift operation or addressing operation is avoided.

The shift operation is a fast and efficient way to convert a byte or word to the filtered pixel value 299.

General Purpose

The lossless compression of the sampled data achieved by the preferred embodiment of the present invention results in high quality video streams that have general purpose application in a number of areas including, without limitation, video conferencing, surveillance, manufacturing, rich media advertising, and other forms of video transmission, storage, and processing.

Lossless Nature/No Artifacts

Once the analog signal is sub-sampled and filtered to select a filtered pixel value which eliminates some of the real world defects, the methods of the present invention compress and decompress the data with no irreversible data loss. Unlike JPEG and MPEG, the decompressed image never suffers from artificially induced blocking or smearing or other artifacts that are result of the lossy compression algorithm itself. As a result even a small sub-sample of the image remains clear and true to the perceived quality of the original image.

Superior Features over RHN Format

When compared against the RHN format, the format and methods of the present invention provide a number of advantages, including, but not limited to, faster speed and

smaller size of encoded data, better performance for both medical and typical video images, and a typically closer representation of the original video signal.

Conclusion, Ramification, and Scope

Accordingly, the reader will see that the compression and decompression steps of the present invention provides a means of digitally compressing a video signal in real time, communicating the encoded data stream over a transmission channel, and decoding each frame and displaying the decompressed video frames in real time.

Furthermore, the present invention has additional advantages in that:

1. it provides a means of filtering real world defects from the video image and enhancing the image quality;
2. it allows for execution of both the compression and decompression steps using software running on commonly available computers without special compression or decompression hardware;
3. it provide decompressed images that have high spatial quality that are not distorted by artifacts of the compression algorithms being used;
4. is provides a variably scalable means of video compression; and
5. it provides a means for reducing the space required in a storage medium.

Although the descriptions above contain many specifics, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of the preferred embodiments of this invention. For example, bit ordering can be altered and the same relative operation, relative performance, and relative perceived image quality will result. Also, these processes can each be implemented as a hardware apparatus that will improve the performance significantly.

Thus the scope of the invention should be determined by the appended claims and their legal equivalents, and not solely by the examples given.

Claims: I claim:

1. A method of compression of graphic images which make up a video stream, comprising the steps of:
 - (a) sub-sampling a number of pixel bits from an image selected from said graphic images;
 - (b) run-length encoding repeated instances of said number of pixel bits; repeating steps (b) and (c) until each said number of pixel bits is encoded in an encoded data buffer.
2. The method of claim 1 wherein the image dimensions are less than or equal to 320 by 240.
3. The method of claim 1 wherein said number of pixel bits is one of the set of 3, 4, 5, 8, 9, 12, 15, 16, and 24.
4. The method of claim 3 wherein said number of pixel bits is extracted from the most significant bits of each color component.
5. An encoded video signal comprising a series of said encoded data buffers.
6. A storage medium in which the encoded video signal as claimed in claim 5 is stored.
7. A method of decompressing an encoded video signal, comprising the steps of:
 - (a) reading a stream of run-length encoded codes;
 - (b) determining a series of pixels based on the values and run-lengths of said codes;
 - (c) combining said pixels into an image; and
8. The method of claim 7 further comprising the step of displaying a series of said images.
9. The method of claim 7 wherein the width and the height of said image are less than or equal to 320 by 240, respectively.

10. The method of claim 7 wherein said codes represent the combination most significant bits of each of the color components of each pixel.
11. A machine for compressing of a plurality of video frames which make up a video signal, comprising
 - (a) a video digitizer configured to digitizing a frame from said video frames;
 - (b) a video memory which is able to receive a plurality of pixels from said video digitizer;
 - (c) a encoding circuit for counting repeated instances of a pixel value when scanning said plurality of pixels and outputting a series of encoded data comprising a combined run-length field and a data field.
 - (d) a memory which is able to store said encoded data;
 - (e) an input/output device.
12. The machine of claim 11 wherein said encoding circuit variably selects one of a set of 3, 4, 5, 8, 9, 12, 15, 16, and 24, as the number of pixel bits.
13. The machine of claim 12 wherein said pixel value is extracted from the most significant bits of each color component.
14. The machine of claim 11 wherein said input/output device is a storage medium.
15. The machine of claim 11 wherein said input/output device is a communications transmission channel.
16. A machine for decompressing an stream of encoded data that represents a video signal, comprising:
 - (a) an input/output device for reading said stream of encoded data;
 - (b) a decoding circuit which can decode the encoded data and output a stream of pixel values; and
 - (c) a memory that is able to store an image comprising said stream of pixel values that can be displayed as frames of a video sequence.

17. The method of claim 1 wherein one or more of the settings of width, height, frame rate, brightness, and contrast of said images are variably altered by a receiver of said encoded data.
18. The method of claim 1 wherein said number of pixel bits are variably altered by a receiver of said encoded data.
19. The method of claim 1 further comprising a step of compressing said buffer with a lossless technique known in the art.
20. The method of claim 8 wherein said images are enlarged by stretching prior to said displaying.
21. The method of claim 1 further comprising a step of encrypting said number of pixel bits.

VARIABLY GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (ZLN)

Abstract: Methods, medium, and machines which compress, enhance, encode, transmit, decompress and display digital video images in real time. Real time compression is achieved by sub-sampling each frame of a video signal, filtering the pixel values, and encoding. Real time transmission is achieved due to high levels of effective compression. Real time decompression is achieved by decoding and decompressing the encoded data to display high quality images. Receiver can alter various setting including but not limited to the format for the compression, image size, frame rate, brightness and contrast.

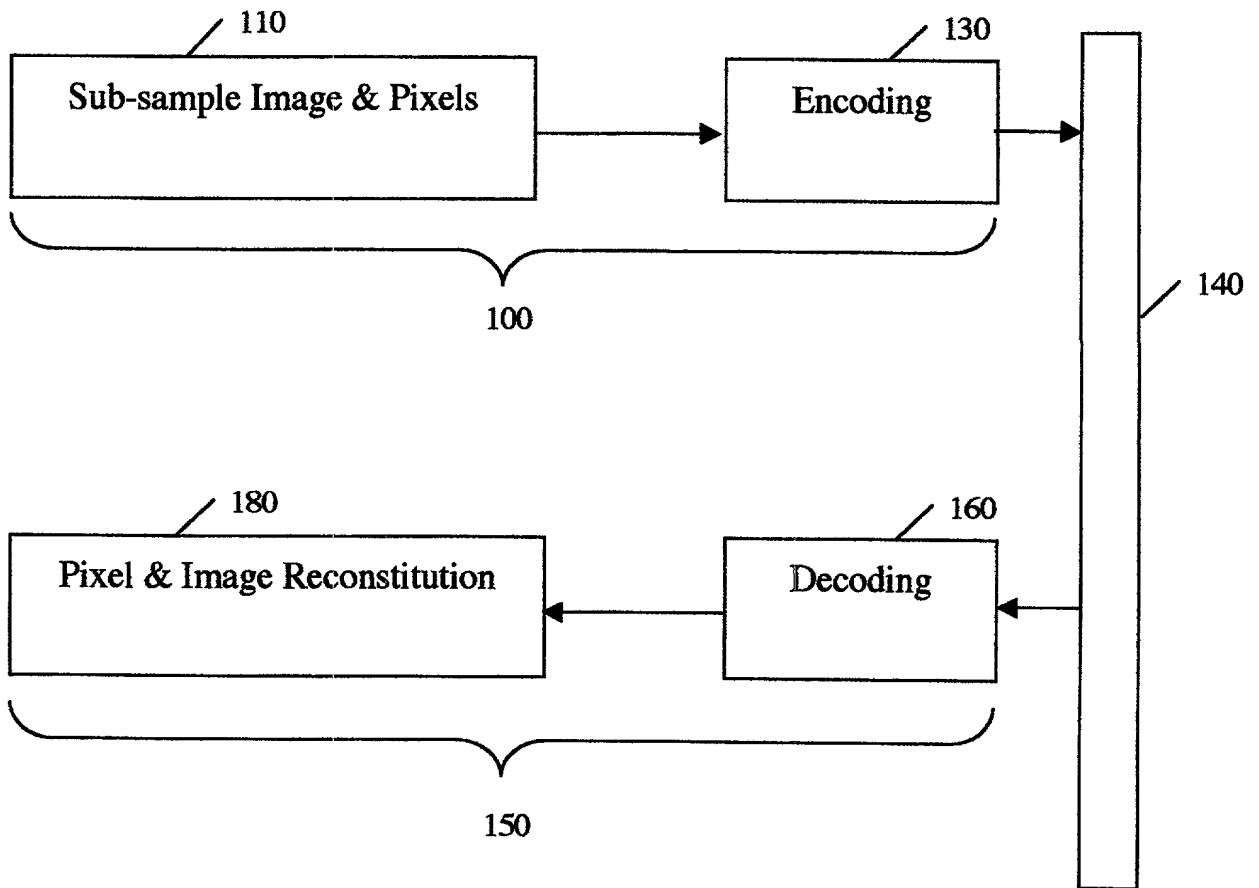


Fig 1

Fig 2A

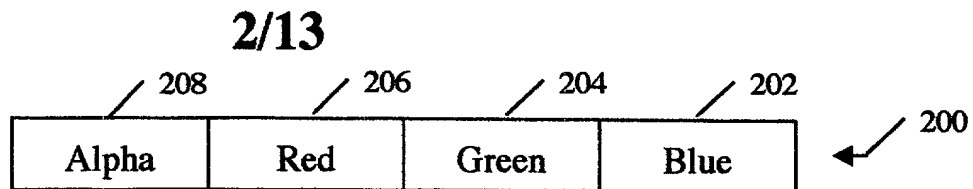


Fig 2B

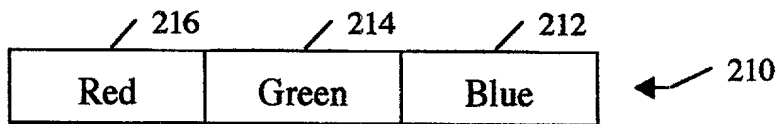


Fig 2C

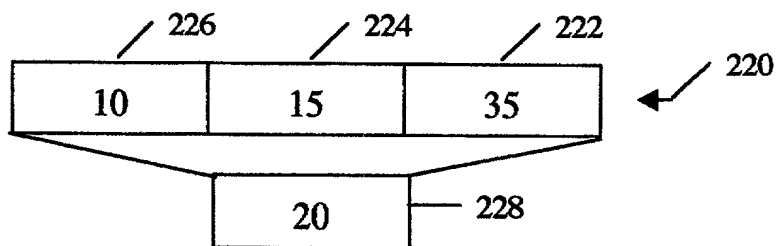


Fig 2D

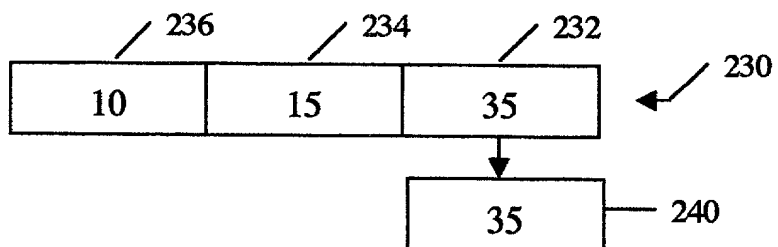


Fig 2E

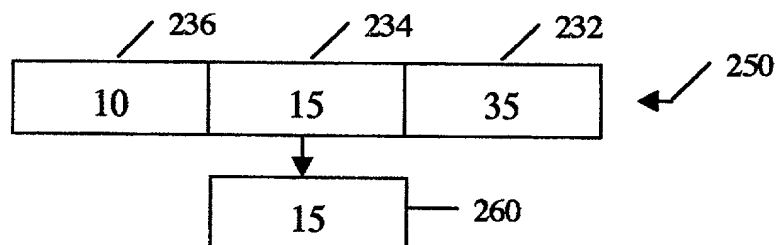


Fig 2F

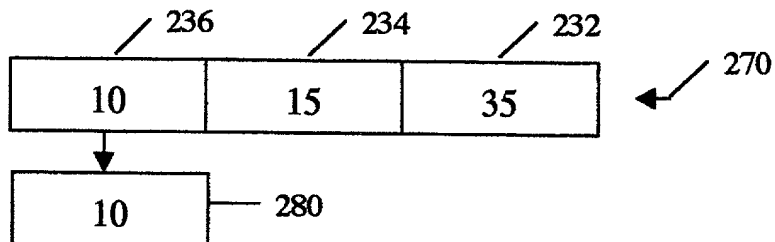


Fig 2G

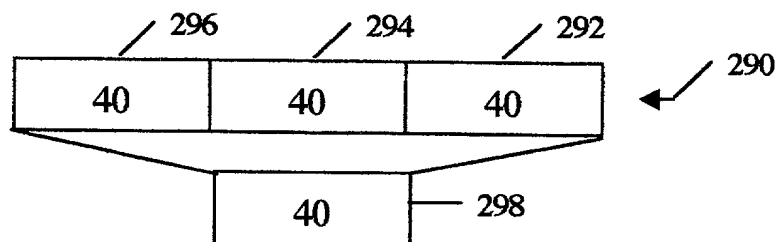


Fig 2H



3/13

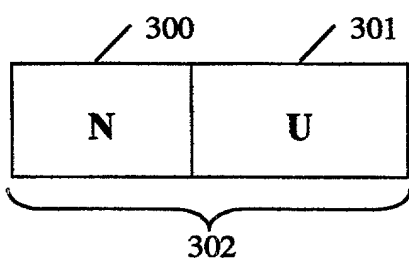


Fig 3A

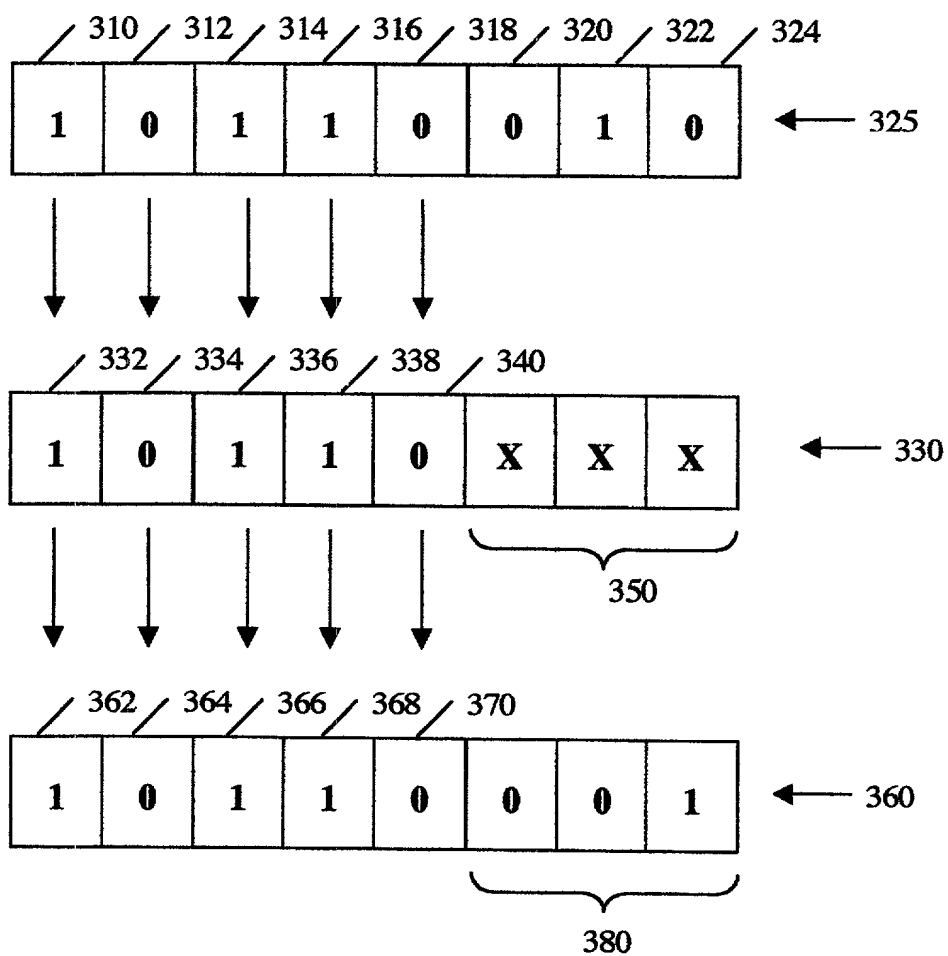
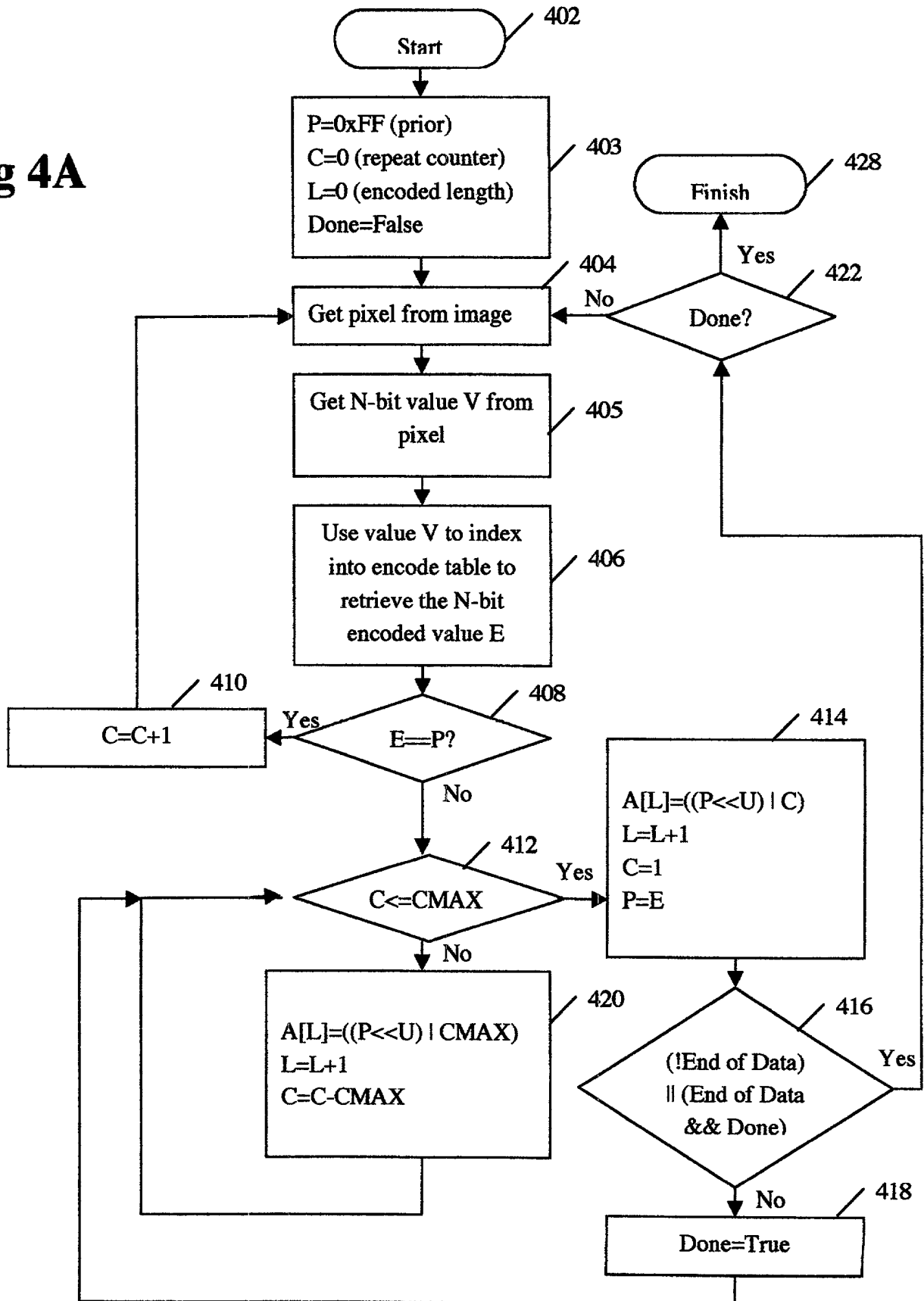


Fig 3B

4/13

Fig 4A



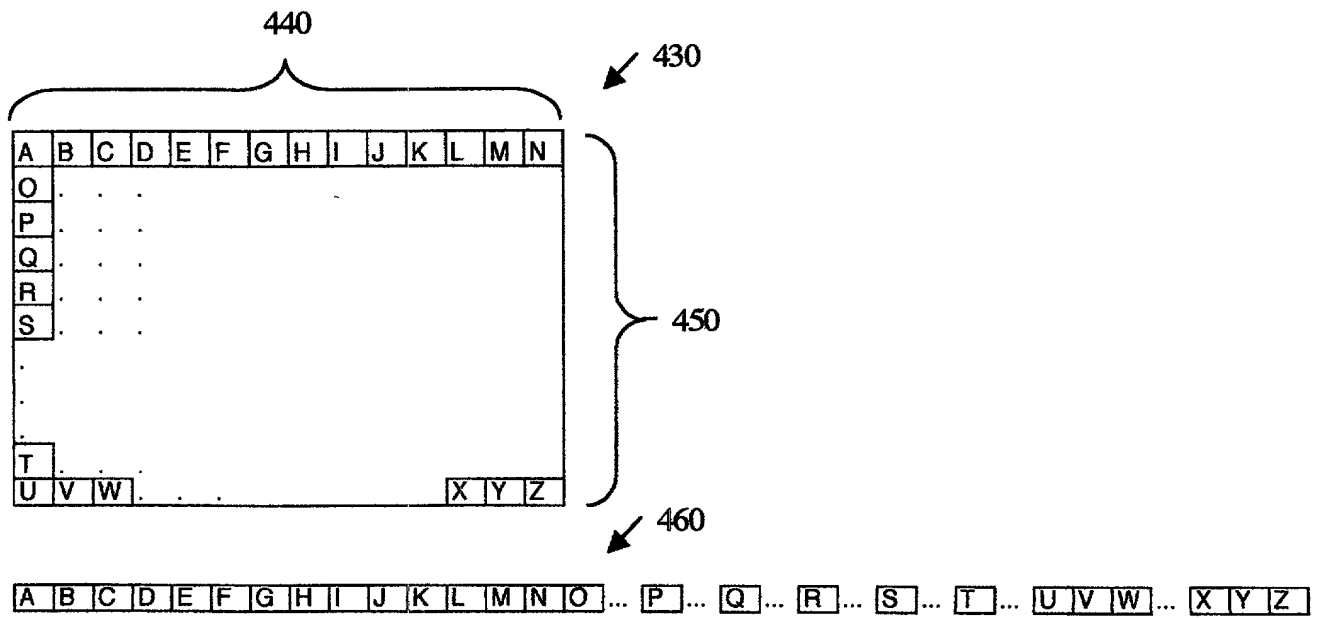


Fig 4B

RHN FORMAT

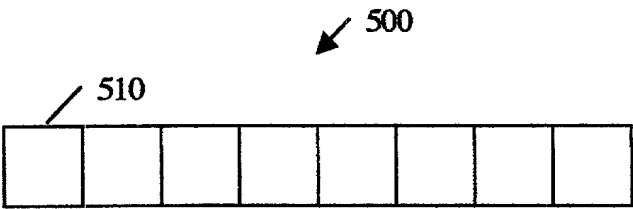


Fig 5A

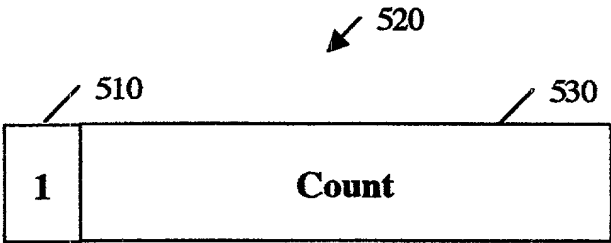


Fig 5B

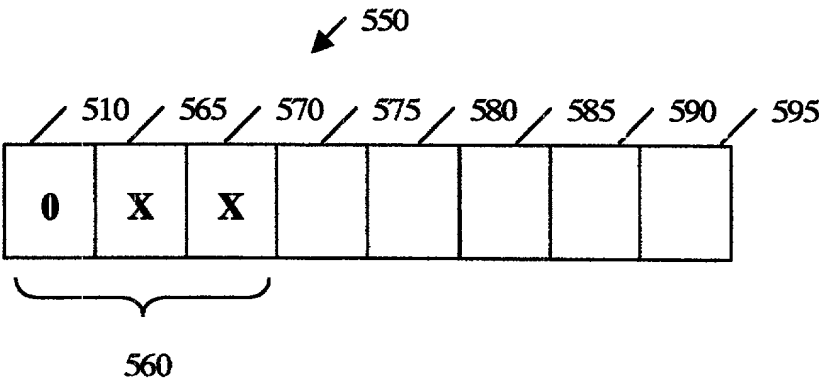


Fig 5C

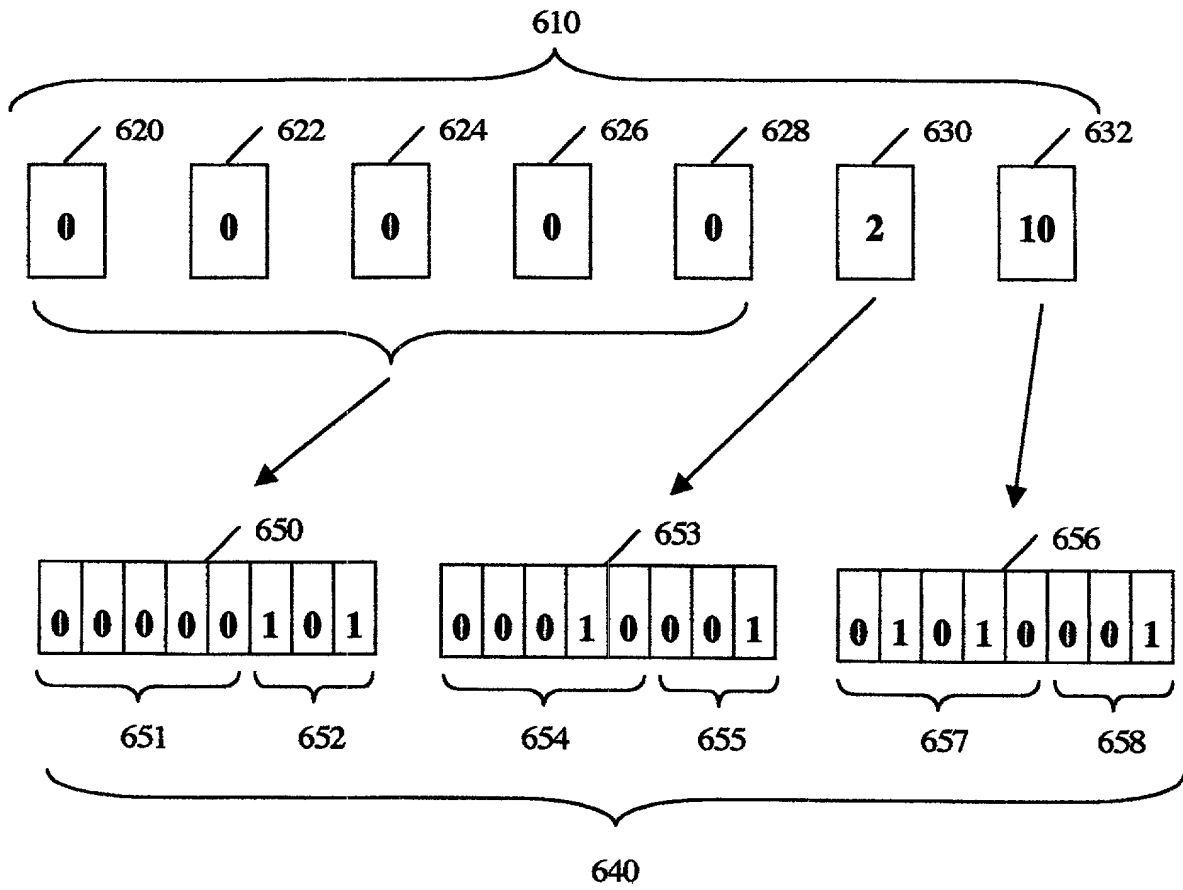


Fig 6

8/13

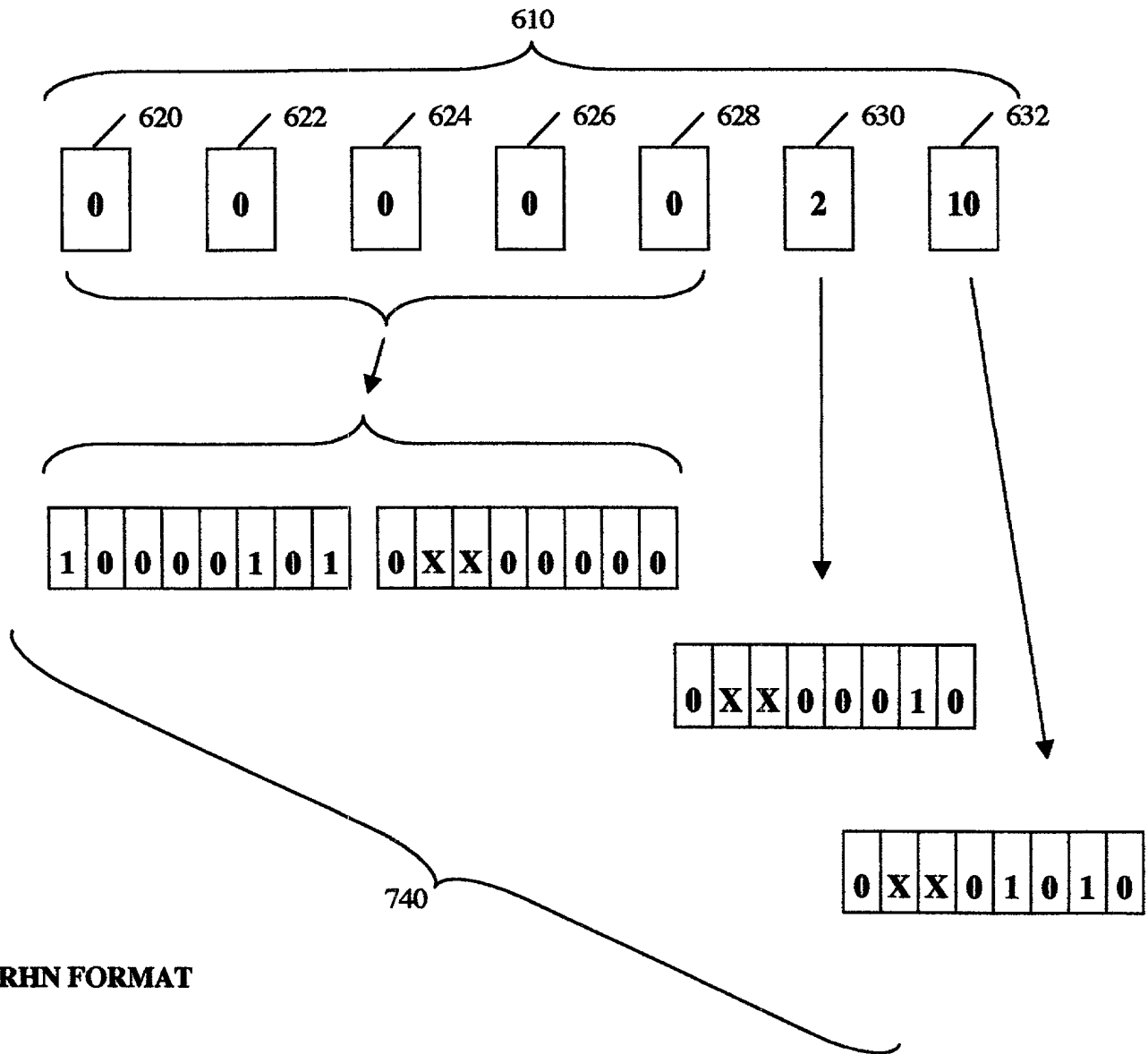


Fig 7

ZL3 (ZL3-5):

D	D	D	C	C	C	C	C
---	---	---	---	---	---	---	---

← 803

ZL4 (ZL4-4):

D	D	D	D	C	C	C	C
---	---	---	---	---	---	---	---

← 804

ZL5 (ZL5-3):

D	D	D	D	D	C	C	C
---	---	---	---	---	---	---	---

← 805

ZL8 (ZL8-8)

D	D	D	D	D	D	D	D	C	C	C	C	C	C	C	C
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

← 808

ZL9 (ZL9-7)

D	D	D	D	D	D	D	D	D	D	C	C	C	C	C	C	C
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

← 809

ZL12 (ZL12-4)

D	D	D	D	D	D	D	D	D	D	D	D	D	C	C	C	C
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

← 812

Fig 8A

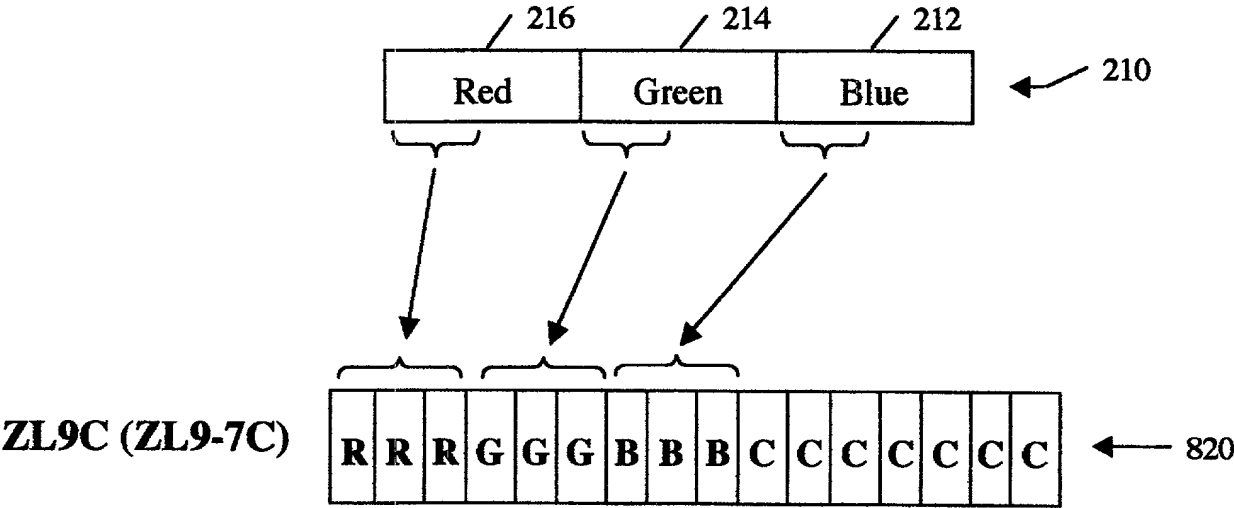
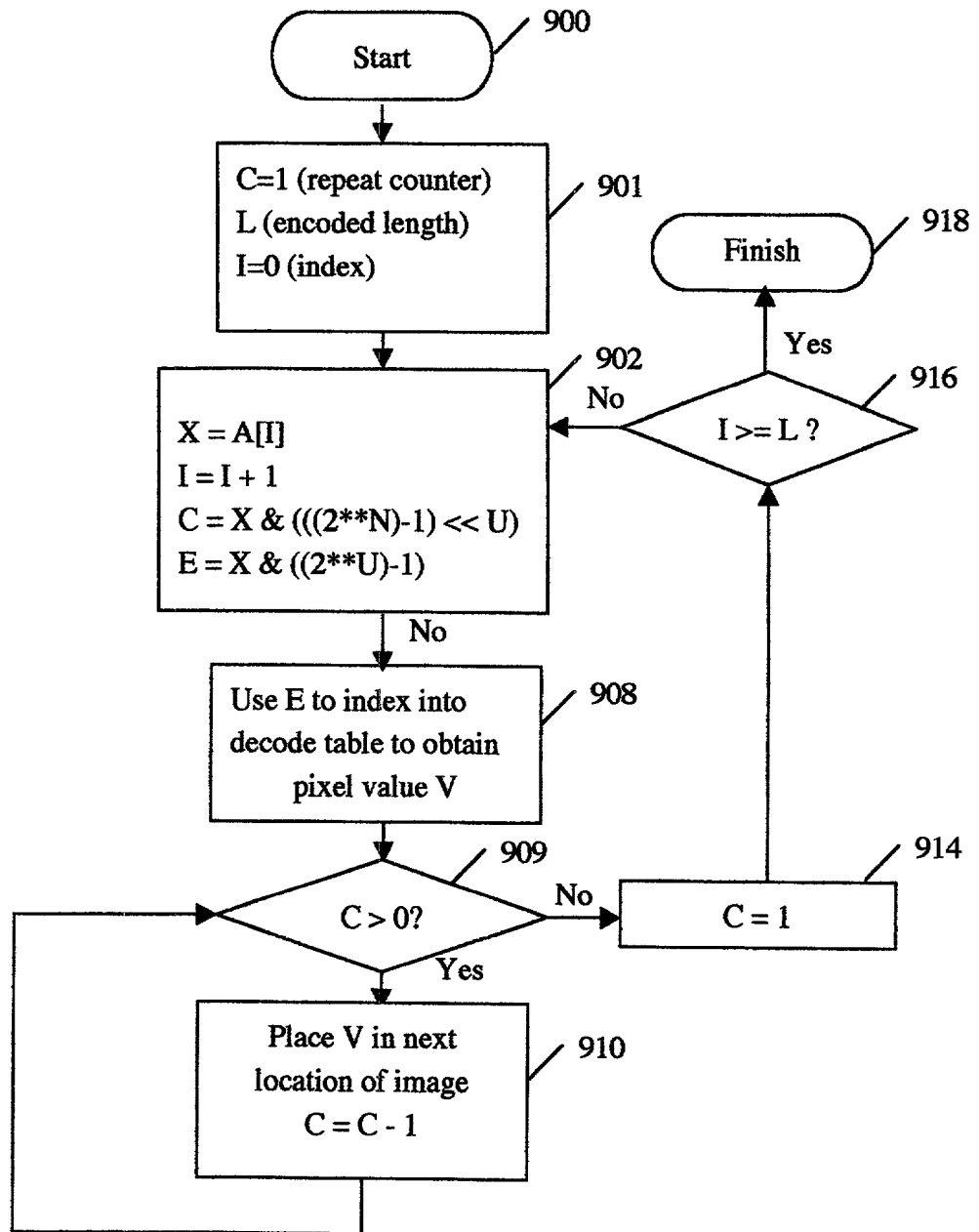


Fig 8B

Fig 9

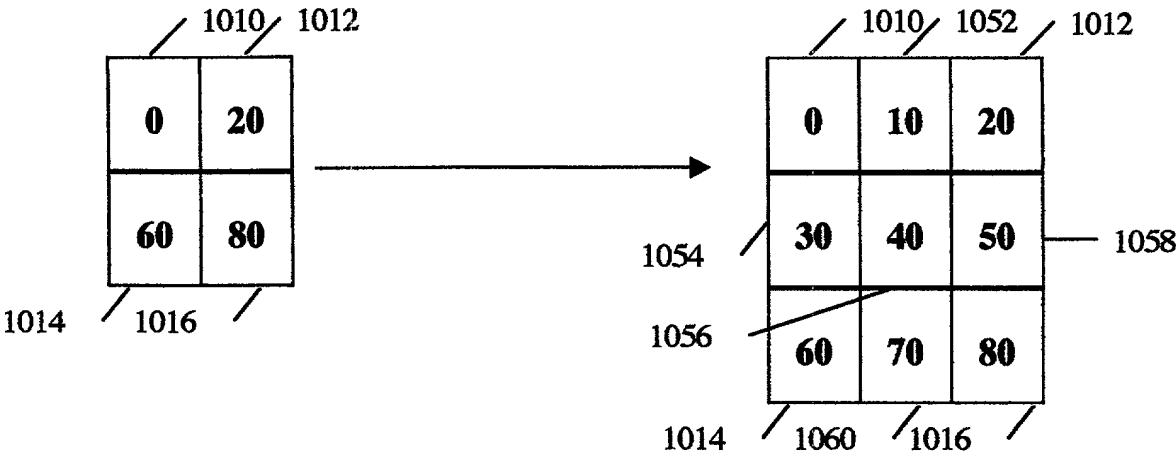




Fig 10

 1100

```
// N=3, W=8
unsigned char encodeTable[ ] =
{
    5, 0, 7, 4, 2, 3, 6, 1
};
```

Fig 11A
 1110

```
// N=3, U=5
int decodeTable[ ] =
{
    0xff << 24 | 1 << (16+U) | 1 << (8+U) | 1 << U, // 0 → 1
    0xff << 24 | 7 << (16+U) | 7 << (8+U) | 7 << U, // 1 → 7
    0xff << 24 | 4 << (16+U) | 4 << (8+U) | 4 << U, // 2 → 4
    0xff << 24 | 5 << (16+U) | 5 << (8+U) | 5 << U, // 3 → 5
    0xff << 24 | 3 << (16+U) | 3 << (8+U) | 3 << U, // 4 → 3
    0xff << 24 | 0 << (16+U) | 0 << (8+U) | 0 << U, // 5 → 0
    0xff << 24 | 6 << (16+U) | 6 << (8+U) | 6 << U, // 6 → 6
    0xff << 24 | 2 << (16+U) | 2 << (8+U) | 2 << U // 7 → 2
};
```

Fig 11B

Declaration for Patent Application

As a below-named inventor, I hereby declare that my residence, post office address, and citizenship are as stated below next to my name and that I believe that I am the original, first, and sole inventor of the subject matter which is claimed and for which a patent is sought on the invention, the specification of which is attached hereto and which has the following title:

"VARIABLE GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (ZLN)"

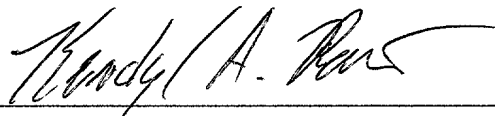
I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to in the oath or declaration. I acknowledge a duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, Section 1001, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

I hereby claim the benefit under Title 35, United States Code, § 119(e) United States provisional patent application Serial No. 60/113,051 filed on 1998 December 21.

Please send correspondence and make telephone calls to the Sole Inventor below.

Signature: Sole Inventor: _____



Print Name: Kendyl A. Román

Date: 1999 December 20

Legal Residence:* Sunnyvale, CA

Citizen of: USA

Post Office Address: 730 Bantry Court, Sunnyvale, CA 94087-3402

Telephone: (408) 739-9517